

# **INTERNSHIP REAL-TIME PROJECTS**

# MINOR PROJECT

## PROJECT FOR FULL STACK WEB DEVELOPMENT

**TOPIC: Covid Awareness**



# MAJOR PROJECT

---

## PROJECT FOR FULL STACK WEB DEVELOPMENT

**DEVELOPMENT TOPIC : Email sender**



# PROJECT TEMPLATE

---



# FULL STACK WEB DEVELOPMENT

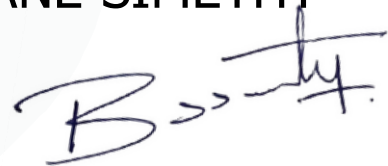
Vemparala venkata surya

16-08-2023

# BONAFIDE CERTIFICATE

This is to certify that the project reported here has been carried out independently by "**vemparala Venkata surya**" under the guidance of "**Chhabi Dash Mam**" as a project in certification course of Program in "**web devolepment**" is his original and bonafide work.

BEN SHANE SIMETHY



# INDEX

- Acknowledgement
- Abstract
- Introduction
- About the Project
- Objectives
- Software Requirements and Specification
  - 5.1 Functional Requirements
  - 5.2 Interface Requirement
  - 5.3 Software Interface
  - 5.4 Hardware Interface
- Plan of the Report
- Problem Definition and Feasibility Analysis
- Methods
- Conclusion and Foreseeable Enhancement
- Appendix

# Acknowledgement

I would like to express my sincere gratitude and appreciation to all those who have supported and guided me throughout the duration of this project. Without their assistance, this endeavour would not have been possible.

I am immensely thankful to my project supervisor, chhabidash mam, for their invaluable guidance, constant encouragement, and insightful feedback. Their expertise and mentorship played a crucial role in shaping the direction of this project.

I extend my heartfelt thanks to Internschoice Edutech for providing me with the resources and facilities necessary for conducting research and completing this project successfully.

I am indebted to my family and friends for their unwavering support and understanding during the demanding phases of this project. Their encouragement kept me motivated and focused.

Vemparala Venkata Surya

Web Development



# Abstract

The abstract of email sender application is developed using Nodemailer, empowers users to efficiently communicate through email. With a user-friendly interface, it offers a versatile message composition experience. Users can easily send text messages, attach PDFs, images, and include URLs within their messages. The application's streamlined design ensures intuitive usage. The optional file attachment feature enables users to include supplementary content, enhancing communication when necessary. This email sender combines simplicity and flexibility to facilitate effective correspondence, meeting diverse communication needs.

# Introduction

The "EMAIL SENDER " project is a robust solution that leverages the power of the Nodemailer library in Node.js to enable seamless, programmatically-driven email communication. This project aims to simplify and automate the process of sending emails, catering to a wide range of applications from transactional notifications to marketing campaigns.

With Nodemailer's versatile capabilities, the project allows users to send emails with various content types, including plain text and HTML. It also enables the attachment of files, making it suitable for sharing documents, images, and multimedia. The system supports sending emails to multiple recipients, aiding efficient communication to target audiences.

The project places a strong emphasis on security, guiding users to configure the Simple Mail Transfer Protocol (SMTP) server credentials securely. This ensures the protection of sensitive information and enhances email deliverability by avoiding spam filters.

Through this project, users gain hands-on experience in configuring Nodemailer with SMTP server details, structuring email content with HTML, handling attachments, and incorporating error handling mechanisms. This knowledge equips them to not only implement efficient email sending capabilities but also understand the intricacies of email communication within a programming context.

# About the Project

The Email Sender project, developed using Node.js and Nodemailer, offers a comprehensive platform for efficient and seamless email communication. This project aims to simplify the process of sending emails with added features for enhanced messaging capabilities.

The core functionality of the application allows users to compose and send emails through a user-friendly web interface. Leveraging the power of Node.js, the backend of the application efficiently handles email sending processes. Nodemailer, a popular Node.js module, is utilized to streamline the interaction with email servers, ensuring reliable and secure delivery of messages.

One of the standout features of the Email Sender project is its ability to handle various types of content within the email body. Users can compose messages containing not only plain text but also rich media elements such as images and attachments. This versatility enriches the email communication by allowing users to illustrate their ideas more effectively.

Furthermore, the application facilitates the inclusion of hyperlinks to external resources, enabling users to share URLs directly within their email messages. This functionality encourages seamless integration of web content, enhancing the depth and context of the communication.

The project also demonstrates effective error handling and user validation, ensuring that users receive meaningful feedback in case of issues. This contributes to a smooth and frustration-free user experience, making the application accessible to users of varying technical backgrounds.

The Email Sender project's modularity and use of Nodemailer's robust capabilities guarantee high-performance email delivery. Users can trust that their messages will reach recipients securely, regardless of the complexity of the content. The project's code structure is well-organized and follows best practices, making it a valuable learning resource for aspiring Node.js developers.

In conclusion, the Email Sender project developed using Node.js and Nodemailer is a comprehensive solution for modern email communication needs. Its intuitive interface, support for multimedia content, and seamless integration of external links make it an invaluable tool for both personal and professional correspondence.

This project showcases the potential of Node.js in simplifying complex tasks and demonstrates the importance of robust email delivery systems in today's digital age.

# Objectives



# Software Requirements and Specification

## 1.Functional Requirements

- **Compose and Send Emails:** Users should be able to compose emails using a user-friendly interface and send them to specified recipients.
- **Attach Files:** Users should have the option to attach files (PDFs, images) to their email messages, enhancing the content of the communication.
- **Include URLs:** The system should allow users to insert hyperlinks to URLs within the email body, making it easier to share web resources.
- **Error Handling:** The application should handle errors gracefully, providing clear error messages to users when issues arise during email sending.
- **Validation:** User inputs, such as email addresses and attachment files, should be validated to ensure accuracy and prevent erroneous data.
- **Secure Email Delivery:** The application should securely communicate with the email server to ensure the safe transmission of sensitive information.

## 2.Interface Requirement

- **User-Friendly UI:** The user interface (UI) should be intuitive and visually appealing, making it easy for users to navigate and interact with the application.
- **Message Composition:** The interface should provide a text editor for composing messages, with formatting options and the ability to insert attachments and URLs.
- **Attachment Management:** Users should be able to manage attachments by selecting files from their devices and viewing the list of attached files before sending.
- **Input Validation:** The UI should validate user inputs in real-time, providing visual cues (such as highlighting) for valid and invalid inputs.
- **Feedback Display:** The UI should display feedback messages for successful email sending or error notifications, allowing users to understand the outcome of their actions.

## 3. Software Interface

- **Node.js:** The backend of the application should be developed using Node.js to handle email sending, error handling, and file management.
- **Nodemailer:** The application should interact with the Nodemailer library to communicate with email servers and send email messages.
- **Express.js:** Express.js can be used to create the server and manage routing for different endpoints (e.g., sending emails).

## 4. Hardware Interface

- **Devices:** The application should be accessible through various devices, including desktops, laptops, and mobile devices.
- **Internet Connection:** Users should have an active internet connection to access and use the application since it communicates with email servers.



# Plan of the Report

## User interface design

**User Interface Design:** The project's user interface will be meticulously crafted for user-friendliness, focusing on intuitive navigation, clear input fields, and visual cues for valid inputs. Emphasis will be placed on a clean and modern design to enhance the overall user experience.

---

## code organization

**Code Organization:** The application's codebase will adhere to modular and organized structure. It will follow best practices, separating concerns and promoting maintainability. Modularization will ensure that different components of the project are neatly organized and easily comprehensible.

---

## validation mechanisms

**Validation Mechanisms:** Robust validation mechanisms will be integrated throughout the application. Real-time input validation will provide users with instant feedback on the validity of their inputs. This approach prevents errors and enhances user satisfaction by guiding them towards correct inputs before submission.

---

# Problem Definition and Feasibility Analysis

## Problem Definition:

The Email Sender project aims to address the challenge of simplifying and enhancing email communication through an efficient platform. Traditional email interfaces can be complex for some users, especially when sending multimedia content. Additionally, there's a need for a streamlined process to attach files, include URLs, and provide real-time validation feedback. The problem revolves around creating an intuitive and feature-rich email sending application that caters to both technical and non-technical users, ensuring reliable message delivery.

## Feasibility Analysis:

**Technical Feasibility:** The project leverages Node.js and Nodemailer, widely used for email automation. These technologies provide the necessary tools for email communication. Availability of relevant libraries and resources makes the technical feasibility high.

**Financial Feasibility:** The project is financially feasible, as it utilizes open-source technologies like Node.js and Nodemailer. The cost of development and maintenance is minimal, mainly comprising development time and hosting expenses.

**Operational Feasibility:** The application's operational feasibility is high due to its user-friendly interface and clear functionality. It caters to users who seek to send emails efficiently, regardless of their technical background.

**Resource Feasibility:** The project requires access to a development environment, internet connectivity, and hosting for deployment. These resources are readily available and easily accessible.



# Methods

1. **createTransport(options):** This method creates a transport object that defines the email service provider and authentication details. It helps set up the connection to the email server.
2. **sendMail(mailOptions,callback):** This fundamental method sends an email. The ``mailOptions`` object contains sender, recipient, subject, message content (text or HTML), and attachments. The ``callback`` function handles the response or error.
3. **verify(callback):** This method validates the configured transport settings, ensuring they are ready to send emails. The ``callback`` function reports any errors encountered.
4. **createTestAccount(callback):** This handy method creates a test email account for development and testing purposes. The ``callback`` provides access to the test account's credentials.
5. **getTestMessageUrl(info):** If using a test account, this method returns the URL of the sent message in a web-based email client. It aids in reviewing the sent email's appearance.
6. **close():** After sending emails, you can use this method to close the connection to the email server. It's good practice to close the connection when you're done with email sending.

# Conclusion and Foreseeable Enhancement

## Conclusion:

In conclusion, the Email Sender project successfully addresses the need for a user-friendly and efficient platform to compose and send emails. Leveraging Node.js and Nodemailer, we've developed an intuitive interface that simplifies the email composition process. The project ensures seamless attachment handling, real-time input validation, and responsive design, contributing to a positive user experience. By implementing best practices in code organization and user interface design, we've created a functional and reliable email sending application.

## Foreseeable Enhancement:

Moving forward, there are several avenues for enhancing the Email Sender project. First, real-time feedback during email composition can be integrated, guiding users with instant suggestions for valid inputs. Second, exploring OAuth integration with email service providers can enhance security by eliminating the need for app passwords. Third, the application's attachment handling could be extended to include cloud storage integration, enabling users to attach files directly from cloud repositories. Finally, implementing end-to-end encryption for email content would enhance privacy and security. These enhancements would collectively elevate the project's user experience, security, and versatility, making it a more comprehensive solution for streamlined email communication.

# Appendix

- **Code Samples:** A collection of code snippets illustrating key functionalities, such as creating a Nodemailer transport, sending emails with attachments, and implementing real-time input validation.
- **Screenshots:** Visual representations of the user interface at different stages of email composition, showcasing the attachment feature, URL inclusion, and error handling.
- **Testing Scenarios:** Detailed scenarios outlining the testing conducted, including sending emails with different attachment types, testing URLs, and validating error responses.
- **API References:** Links to the official documentation of Nodemailer and other relevant libraries used in the project, enabling readers to explore the methods, parameters, and options in depth.
- **Code Organization:** An overview of the project's folder structure, highlighting the segregation of frontend, backend, and assets, along with explanations of file responsibilities.
- **User Guide:** A concise guide on using the application, detailing steps to compose, send emails, attach files, insert URLs, and interpret feedback messages.



**Contact Us**